1-1-1984

A computer controlled milling machine.

Avinash Anand Kenkare

A COMPUTER CONTROLLED MILLING MACHINE

by

AVINASH ANAND KENKARE

A Thesis

Presented to the Graduate Committee

of Lehigh University

in Candidacy for the Degree of

Master of Science

in

Mechanical Engineering

Lehigh University

1984

A COMPUTER CONTROLLED MILLING MACHINE

by

AVINASH ANAND KENKARE

A Thesis

Presented to the Graduate Committee

of Lehigh University

in Candidacy for the Degree of

Master of Science

in

Mechanical Engineering

Lehigh University

1984

CERTIFICATE OF APPROVAL

This Thesis is accepted and approved
in partial fulfillment of the requirements
for the degree of
Master of Science
in
Mechanical Engineering

5/10/84
(date)

Professor in Charge

Chairman of Department

ACKNOWLEDGMENTS

I would like to express thanks to Dr. Mark Lang for his constructive suggestions, encouragement, and support throughout the course of this study. I must also express my special appreciation to Mr. Jim Bunderla for his excellent cooperation and workmanship as regards the fabrication of the Milling machine.

I am indebted to my mother, Mrs. Rekha A. Kenkare for the unwavering support and motivation she provided me in order that I sucessfully complete this work.

Table of Contents

LIST OF FIGURES

ABSTRACT

At the design/concept stage in the manufacture of a part, the designer makes a variety of descisions, and arrives at a part model which he finds suitable. Now, all the data necessary for the manufacture of a part such as material, dimensions etc. are part of the design. Hence, there is no need for this data to be input again, at the time of manufacture of the part.An attempt has been made to recognize this fact, and make use of it. An essential element of this process is a machine capable of responding to digital input.

Once a part has been designed on a CAD system like UNIGRAPHICS , and a simulation of the NC machining has been performed, a data file known as the Cutter Location Source File (CLSF) is produced. All data essential for the manufacture of a part is contained in this file. A post-processor (POST) has been written to process the data contained in the CLSF, and generate an output file that can be input to a computer controlled milling machine. This output of the POST is machine dependent. A machine capable of 3 axis machining whoose actuators are D.C. stepping motors has been designed. The machine is relatively light, and it's chief purpose is to serve as a tool for learning. It is designed to machine material such as proofboard, and

balsa wood. A controller for this machine has been built. This includes an interface linking the actuators to a mini-computer (D.E.C.'s MINC) , and software capable of two and a half axis machining. The data generated by the POST is read by the controller which in turn induces motion in the actuators thereby machining the desired part. The desired part is therefore manufactured simply by passing the data contained in the design of the part thru a series of data processors (The POST and The Controller).

Chapter I

INTRODUCTION

There are several advantages to having a computer controlling the process of feeding numerical data to a manufacturing tool such as a milling machine in addition to the basic advantages of numerical control such as accuracy and speed. Manufacturing is " a monolithic indivisible process of data processing applications ". When the designer's job is over, virtually all the data necessary to manufacture a particular part is present. Factors such as feed rate and tool speed also can be calculated based on the material and type of " cut ", ie. a straight line or curve. If the designer makes use of a computer aided design system, there is no need for all the data input in the design process to be re-calculated. This is the most important benefit of having a computer controlled milling machine. All one has to do is process the data appropriately in order to manufacture the necessary part. Since a computer can be used to do the processing job, there is absolute computer control in manufacturing the part. If a simulation of the machining process is also used, then costly mistakes can be avoided. Changes are also made very easily, since all that has to be done is that at some stage in the processing operation data has to be appropriately modified.

A brief look at Numerical Control at this stage is important for obtaining the proper perspective on the subject of computer control. The difference between numerical control of a machine and conventional control is that a human operator does not guide the tool but this is done by an electronic unit that induces motion. A tape containing data is fed into the electronic unit, and the data is interpreted by the unit (using solenoids) to do the machining. Besides the major advantages mentioned before, there are other advantages such as reducing the likelihood of operator error, reduction of the number of parts that are inspected, and more capability of manufacturing complicated parts.

In order to benefit from these advantages a numerically controlled machine has to be better designed and better constructed in almost every respect than a standard one. No longer can the human operator be counted upon to "nudge" a handwheel for precise positioning or to compensate for the few thousandth of cocking in a spindle that he has learned to expect from a given machine. Regardless of the level of sophistication of the control system ,it cannot compensate for the expertise of an experienced machinist. An operator of a NC machine cannot be expected to correct for lost motion or deflections (basic errors) etc. Also, certain machine elements , such as screws and gearing, are an

integral part of the servo system which is relied upon to accurately position the machine. Thus, an NC machine must be stronger, tighter, and more accurate originally. Maintenance too must be good.

Machine performance is designed into a machine. Once a machine is built, there is relatively little that can be done to significantly improve it. Therefore, sufficient care must be taken to design the machine, keeping in mind the requirements that it has to meet upon completion. There are numerous factors in the condition and construction of the machine that will affect the machine's performance during operation. The foremost among these are backlash, vibrations, deflections in the feed drives, friction, stickiness, etc.

Before one looks at computer control of a milling machine, it is important that one looks at manual programming in order to obtain a better perspective on CNC machining.

In manual programming, the operator sits down and computes the points that the machine has to access manually (usually using a calculator), and then he/she painstakingly goes over them to make sure that there is absolutely no error in them. Finally, he punches out a paper tape which he feeds into his NC machine.

There are many disadvantages to this type of approach:

1) Non-conservation of data:

There is every possibility that all the data the designer has so painstakingly made a part of his design will not be used to the fullest advantage, and in fact, even vital data may be left out.

2) Possibility of human error:

Since there is a significant amount of human interaction, this possibility is very high.

3) Slow:

It takes hardly any time for a computer to process data compared to the amount of time a human operator might take, hence time spent in NC machining as compared to CNC machining is a lot more.

4) Changes are difficult to implement:

After a tape has been punched out, changes(due to mistakes or design modifications) are difficult to implement. The entire original tape may have to be discarded, and a new one may have to be punched out.

5) Unreliable interface:

The interface for a conventional NC machine consists of solenoids on a paper tape reader that respond to the punched tape. This interface is not too reliable, and if the tape's

punching is even slightly below specifications, the reader is prone to make a mistake.

6) Tedious:

This entire task of calculating points, punching out tapes, etc. is very laborious and time consuming.

At Lehigh, there exists a UNIGRAPHICS Computer Aided Design package developed by McDonnel Douglas Automation Corporation which supports an NC machining package. This package includes a simulation of the machining process. A data base is produced by this simulation which contains all the information necessary to implement the milling process. A link is established between the creation of a part and it's manufacture. This will help students at Lehigh, and others to get a better understanding of the process of integrating CAD/CAM.

Complete computer control of the milling of a part implies being able to go from a concept (which is physically feasible and manufacturable) to the actual machining of the part without a human interface.

A CAD system (like UNIGRAPHICS) is used to create the part along with it's physical dimensions on a graphics terminal. Once the user is satisfied with his creation, this part is stored in the memory of the computer.

The next step in the process is creating a data file that contains all the information necessary to machine the part. On the UNIGRAPHICS system, this can be done by simulating the machining process. This process of simulation generates a data file called the CL (Cutter Location) file. The file, if accepted by the user, is processed to generate the CLSF or Cutter Location Source File.

The CLSF contains all the data necessary for machining the part. However, the data format is such that it is not acceptable by the machine as input. This data has to be post-processed by using a post-processor (A program usually referred to as a "Post."). A program called " ANAND " was written for this purpose. The data generated by the post-processing is stored in a file known as "Partname".post (in future, referred to as ILL.post).

Before one looks at the process any further, it is important to take a look at the milling machine - espescially it's actuators. Motion is possible along the traditional three orthogonal axes viz. X, Y, Z. The three actuators are stepping motors, and hence they need digital inputs in the form of pulses. With the help of an interface, the motors are connected to the mini-computer, viz. DIGITAL's MINC.

Each of the motors drives a ball screw assembly. The XY table consists of three plates. On the bottom two are mounted the stepping motors. On the top plate, the part is to be mounted. On top of the plate that moves only in the X direction is a stepping motor that moves the top plate in the Y direction. Thus, the top plate moves in both the X and Y directions.

Now, the tool itself is moved up or down in the Z direction. The vertical motion assembly is supported by an Aluminum beam at a fifty degree angle to the Z axis. A steel beam is used to damp out vibrations. This steel beam also links the vertical axis support to the XY table.

Although translational motion is achieved via the ball screws, support for the tool and the plates is provided by ball bushings mounted on rails.

The final step in the process is the controller. This is a program which runs on the MINC and converts the data from ILL.post to the format necessary to drive the stepping motors via the interface. Using a certain feed rate that varies according to the situation, the machine causes the part (stock) to be cut against a constantly turning tool.

As is evident from the above description, the whole process involves using the computer as a tool to process data needed to mill a part.

Chapter II

THE MILLING MACHINE

What follows is a specific descripion of the milling machine fabricated for the purpose of this research.

The need was for a machine that was capable of motion along the 3 orthogonal axes - X, Y, Z. The specifications for motion were :

10 inches by 8 inches in the x,y plane

3 inches in the z plane.

Another design specification involved the size, weight of the machine. The machine had to be " portable "; which meant that two people should be able to move it without much difficulty. Also, since the machine was to be used for classroom demonstrations etc., it's mobility was essential. The machine therefore, could not be mounted to the floor.

The choice of the motors that were to be used in the machine was also severely limited. Motors salvaged from an old machine were to be used. The main reason for doing this was to cut costs. The linear motion had to be reasonably accurate. Only fasteners could be used so as to provide easy assembly, and disassembly. No welding of joints etc. was permitted for this reason. Finally, the machine had to be reasonable in cost.

Keeping in mind the above design specifications, and factors, the machine was built. Figure (2.1) shows the machine.

The material chosen was Aluminum, since this is relatively strong, yet cheap. It is also comparatively light.

The X, Y table:

In addition to the design factors outlined above, there were several that were specific to the X, Y table. The motors were not really chosen, since they were the ones that were already availible. Their size determined the dimensions of most of the other parts. The amount of motion needed along each axis was the other important factor that determined the size of the plates. Also, the rails were to be mounted in such a way that all clearances such as those required for the "X plate", and "Y plate" to clear the stepping motors below them were met. Aluminum blocks were used to provide those clearances. Now, to avoid any vibrations (likely to occur in any milling machine - espescially a light one), rigidity of the machine was very important. Hence, although the supports gave the appearance of being supports like the ones used in bridges, the basic requirement of rigidity was met. This design although not optimum was suitable for our purposes. The primary objective of this effort it must be remembered is not the

design of the best milling machine, but rather the design and control of one to help provide an insight into Computer controlled machining.

The basic design involved 3 plates mounted one on top of the other, such that the top plate could address any point in the x,y plane within limits. The plates were mounted on top of rails for support. Two sets of rails provide support for each plate.

The motion was provided by attaching a ball screw mount in the center of the plate - this moved on a screw which was turned by a stepping motor. The base plate was mounted on top of Aluminum tubes. Pillow blocks were used to support the screw at one end - the other was attached to the shaft of the stepping motor via a coupling. Below the plates that moved, ball bushings were mounted. These rested on a shaft fixed to the rail on the plate below. Other necessary design features included Aluminum blocks mounted on the plates (to provide the necessary clearances), and mounts for the stepping motor (made of Aluminum). Dimensions, and specifications of the various elements are listed below:

Base plate - Material Aluminum; 3/8" by 28.37"

Middle plate - Material Aluminum; 3/8" by 24.87" by 11.00"

Top plate - Material Aluminum; 3/8" by 9.00" by 11.00"

Ball screws - Warner electric - .5" dia. screw - R0505

NOTE: The three following part numbers are associated with the above Ball screw

Wiper kit; Part 8105-101

Flange; Part 8105-448-002

Ball units; Part 8105-448-006

Rails - SR-i rails for .5" dia. shaft

Pillow blocks - for .5" dia. shaft support

Ball bushings -SPB-8-OPN for .5" dia. shaft

Couplings - Love Joy couplings

bushings - brass bushings to fit pillow blocks

Motor mount - refer Figure 2.2

A special mention has to be made of the actuators used viz. stepping motors. As mentioned before, their choice was limited; the motors are Slo-syn stepping motors, model M112-FD09. The torque to speed characteristic is given in Figure 2.3. The diameter of the shaft is .5 inches. Stepping motors were chosen so as to provide easy digital computer control, and also more accurate computer control. On these motors, it is easy to keep track of back lash in the screws. The computer can " directly " control speed of a stepping motor. This means that simply by varying the number of steps per second that are output from the port of the terminal, one varies the speed of the motor.

Tool or Z axis:

Before one begins to look at the specifics of the vertical axis design, it is important to note the various factors that had to be taken into consideration. Since weight or rather the lack of it was an important consideration,the " z hold " (the beam that holds the plate that supports motion along the z axis) was to be designed appropriately. However, the z axis supported the tool,and therfefore rigidity to absorb vibrations was a necessity. The "z hold" chosen was therefore a beam of solid Aluminum. The "z hold" was attached at on end to the base of the mill, by means of four screws. At the other end was the plate, aligned so that the stepping motor resting on it provided motion along the z axis. The plate was attached to the end of the beam by means of four more screws. The appearance of this was of an inclined beam with a weight hanging at it's one end. The base plate was designed so as to accomodate for the inclined beam and it's load. This type of design was appropriate for our purposes.

There were alternative designs availible, but each were rejected for appropriate reasons. One idea was to use a hollow tube to hold the Z plate in position. Although this helped reduce the weight, this would have resulted in vibrations occuring. One more alternative was to avoid having a load at the top of the beam, but instead have the

stepping motor at the bottom moving the entire vertical assembly. This was rejected owing to our limited choice of stepping motors. The third alternative was to add another support beam to the Z hold beam coming from the opposite side. This while strengthening the assembly considerably, would have added to the weight of the mill significantly. The rigidity was thus taken care of instead by a weight used to attach the vertical axis assembly to the rest of the mill. Keeping in mind the major objective indicated earlier, one must realize that options which involved a much more in depth study, and design of the mill were not given serious thought. The machine's being designed in such a way that only fasteners were used, makes it possible for one to improve on the design in the future. Keeping in mind that changes in the machine's structure are always possible in the future, the vertical axis was designed basically independent of the X,Y table.

It is now important to describe how the motion in the z direction was implemented. The actuator was still a stepping motor, although different from the ones used for the X, and Y axes motion. In this case, owing to the size of the plate, only one rail was necessary. Again, motion was provided by a ball screw. A motor mount was designed according to the dimensions of the smaller stepping motor. The ball screw, rail, and ball bushings were of the same

type as before. A tool grip had to be designed that could hold the tool, and be mounted on the ball screw. The stepping motor was also a Slo-syn stepping motor - model M062-FD03. The torque to speed characteristics are shown in Figure 2.4 The reasons for choosing the stepping motor are as given earlier. The smaller motor being much lighter satisfied our purpose of keeping the load at the top low. Dimensions, and specifications of the
various elements are given below:
Tool grip - refer Figures 2.5a and 2.5b.
z plate Material Aluminum; 3/8" by 13.33" by 4.5"
cutout 3.83" by 1.35"
Description of tool: Precise Router 45000 R.P.M.
z hold: Material Aluminum; cross section: 3.05" by 2.65"
angle: 41 degrees
overall length: 33.67"
Part mounting:

Two slits were made along the major length of the top plate, to provide for flexible mounting. Refer Figure 2.6 for exact location of slits. A sliding flange fixed with the use of screws is used to hold the part firmly. Refer Figure 2.7.

The base:

As explained in the design of the mill's vertical axis assembly, this was designed so that the machine's being top heavy was out of the question. Also, any vibrations etc. were to be damped out by making the base out of scrap steel. By doing this, rigidity was obtained. Although weight was added to the mill, it still met the designer's specifications that any two reasonably built people should be able to lift and move the machine.

After manufacture, the important specifications on the machine are as follows:

Limit for motion in the x direction: 10 inches

Limit for motion in the y direction: 8 inches

limit for motion in the z direction: 3 inches

Tool speed: 45000 R.P.M.

Range of feed rates (same for x, and y):

up to 1.0" per second

Machine weight: 100 pounds

The machine was designed with the help of the CAD system at the Computer Aided Design laboratory at Lehigh University. The software used was UNIGRAPHICS, and the terminals were Tektronics 4014 storage tube terminals. The host to the system was D.E.C.'s VAX 11/780 computer. All figures that are attached have been produced on this system

developed by McAuto. A Calcomp plotter was used to reproduce the parts on paper.

The machine was fabricated at the Mechanical Engineering Machine shop of Lehigh University. The machine was built under the experienced, and capable hands of Jim Bunderla. The Machine shop supervisor was Fred Wehden.

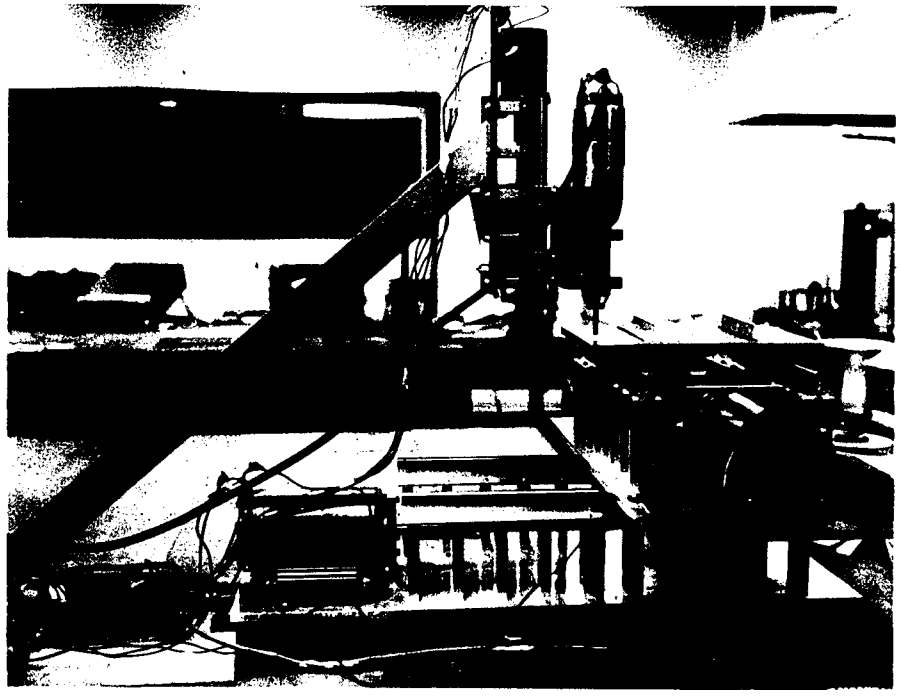
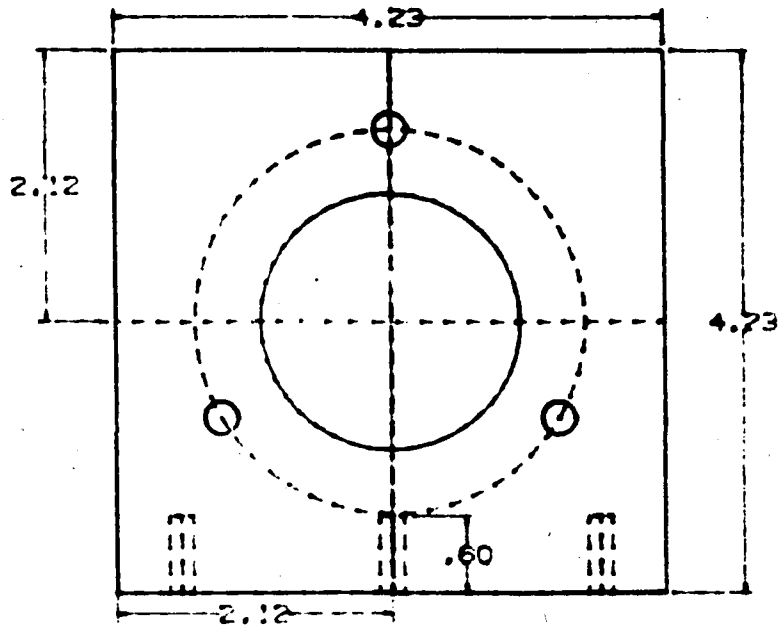
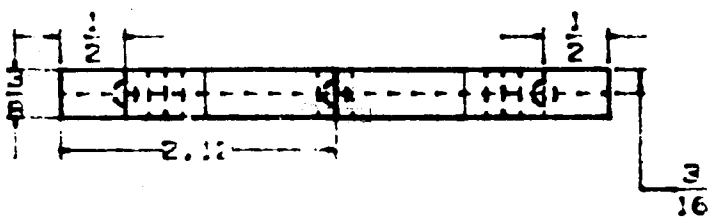


Figure 2.1 The Milling Machine



Top View



Side View

Figure 2.2 XY Motor Mount

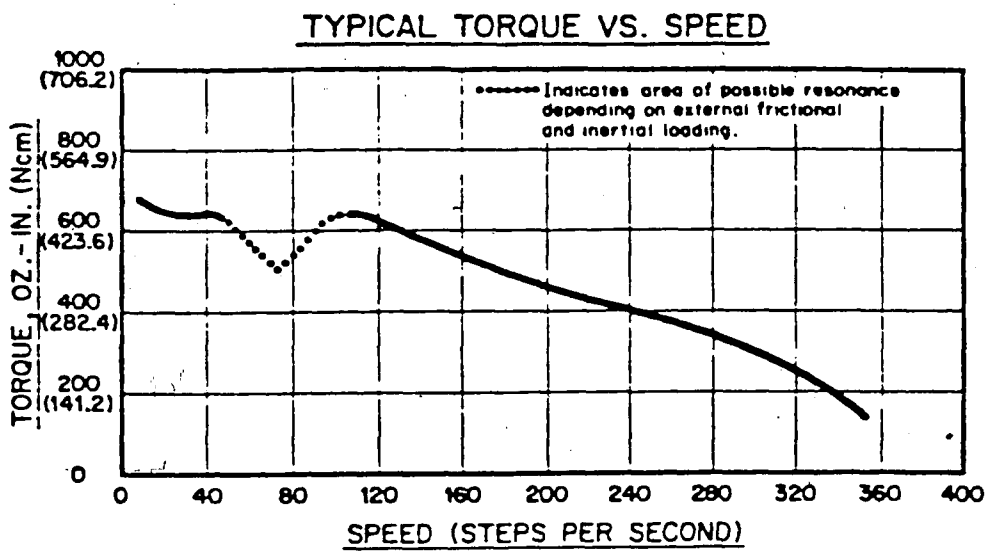


Figure 2.3 Torque-Speed characteristic of X,Y axis motor.

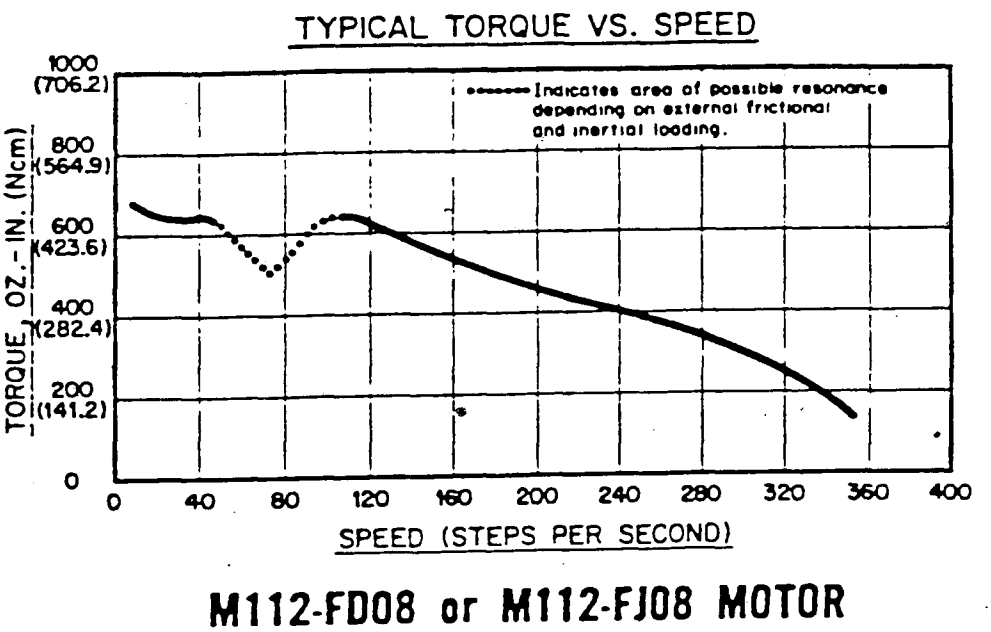


Figure 2.3 Torque-Speed characteristic of X,Y axis motor.

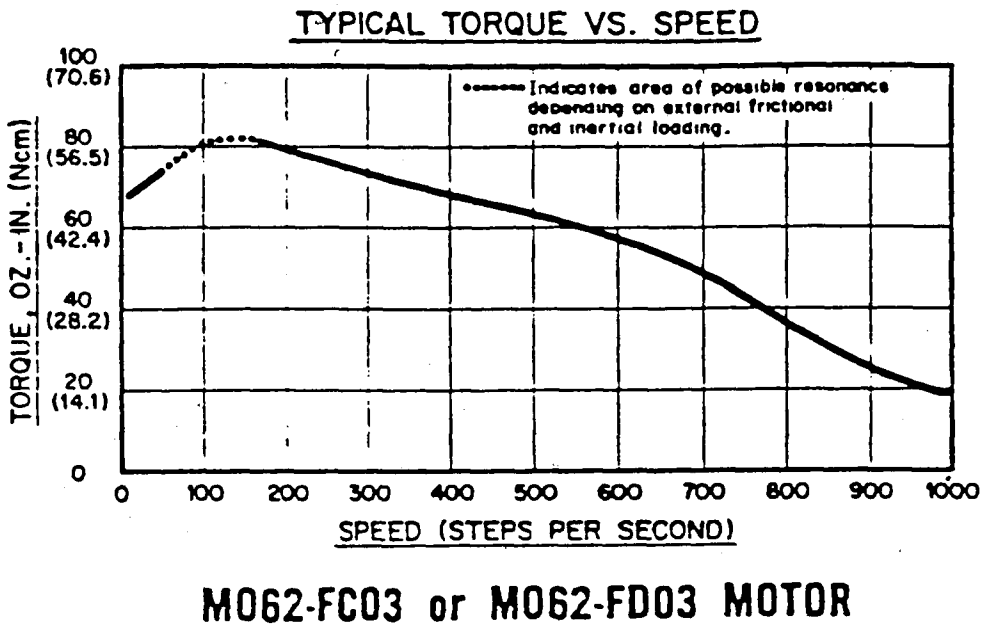


Figure 2.4 Torque-Speed characteristic of Z axis motor.

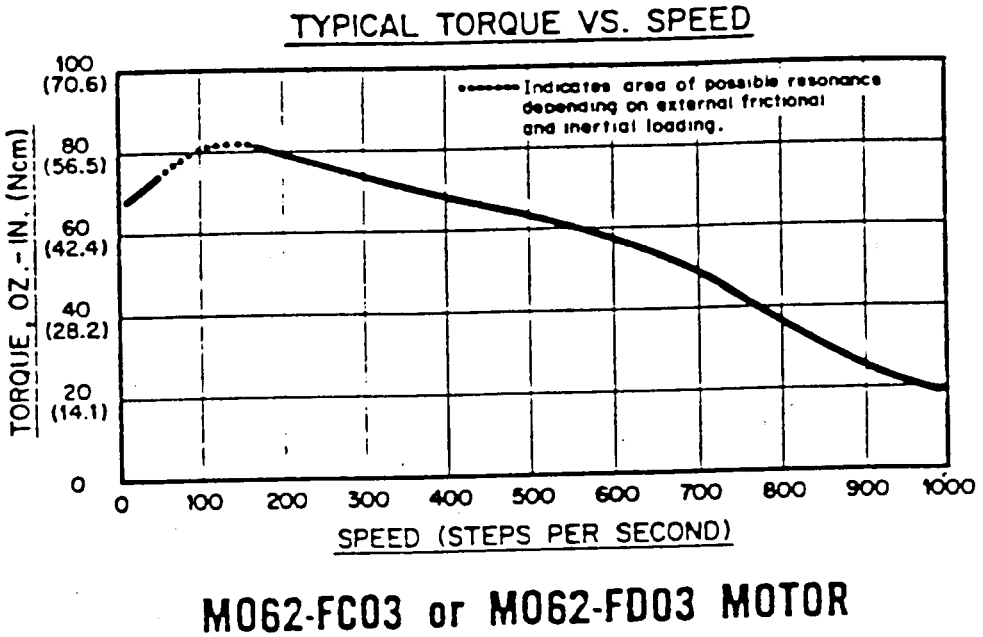


Figure 2.4 Torque-Speed characteristic of Z axis motor.

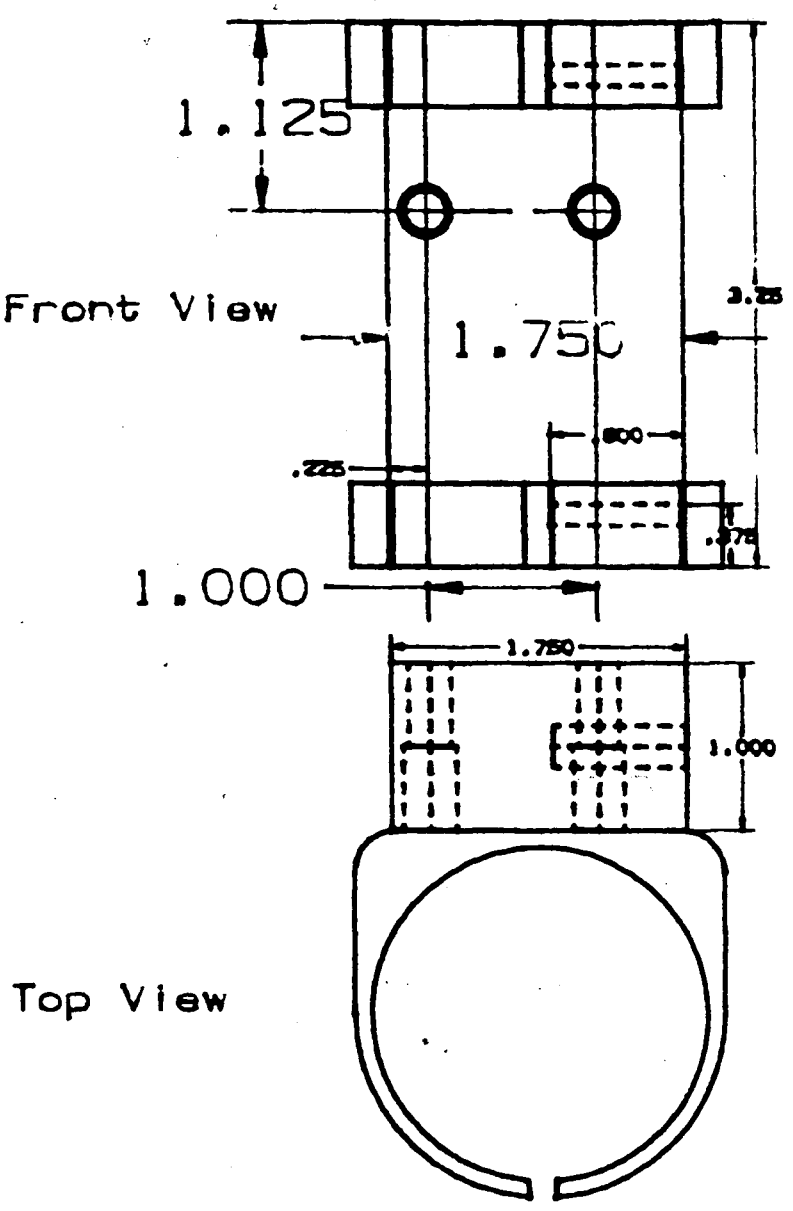
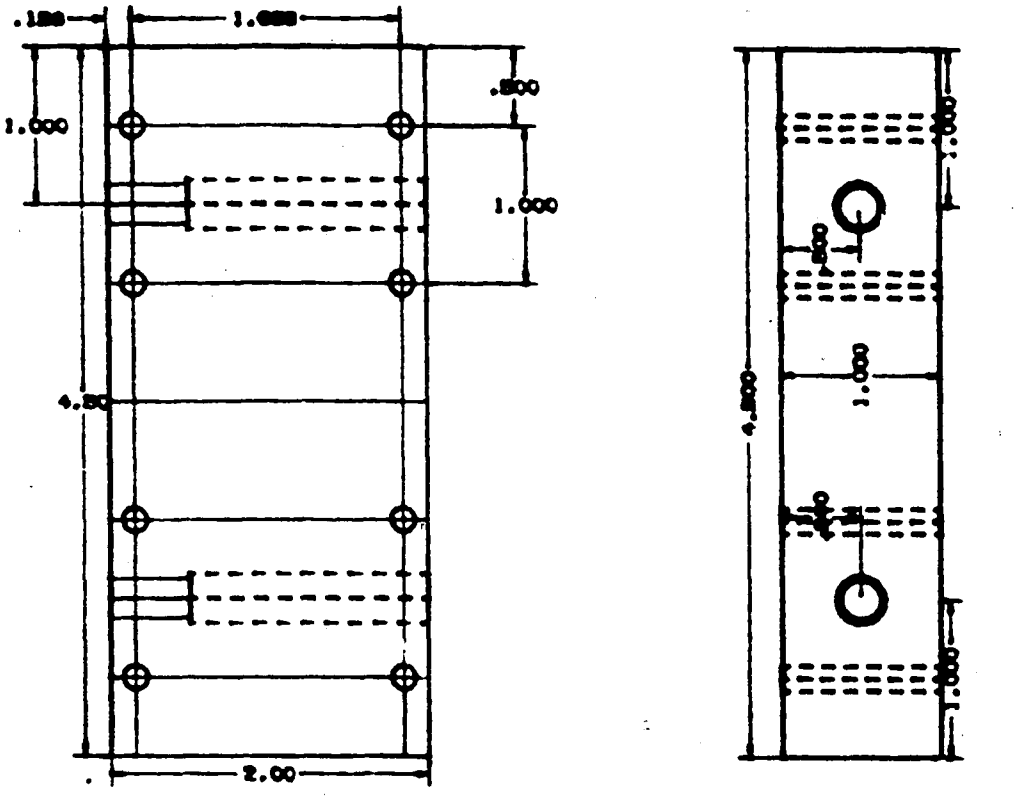


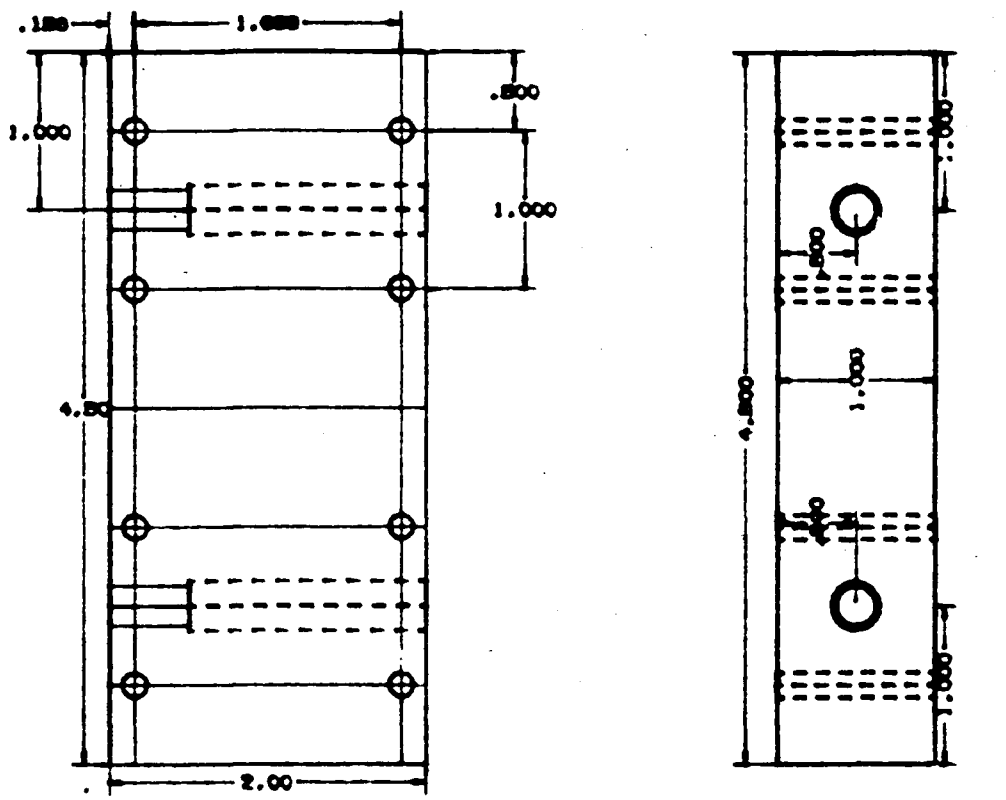
Figure 2.5a Tool Grip



Top View

Side View

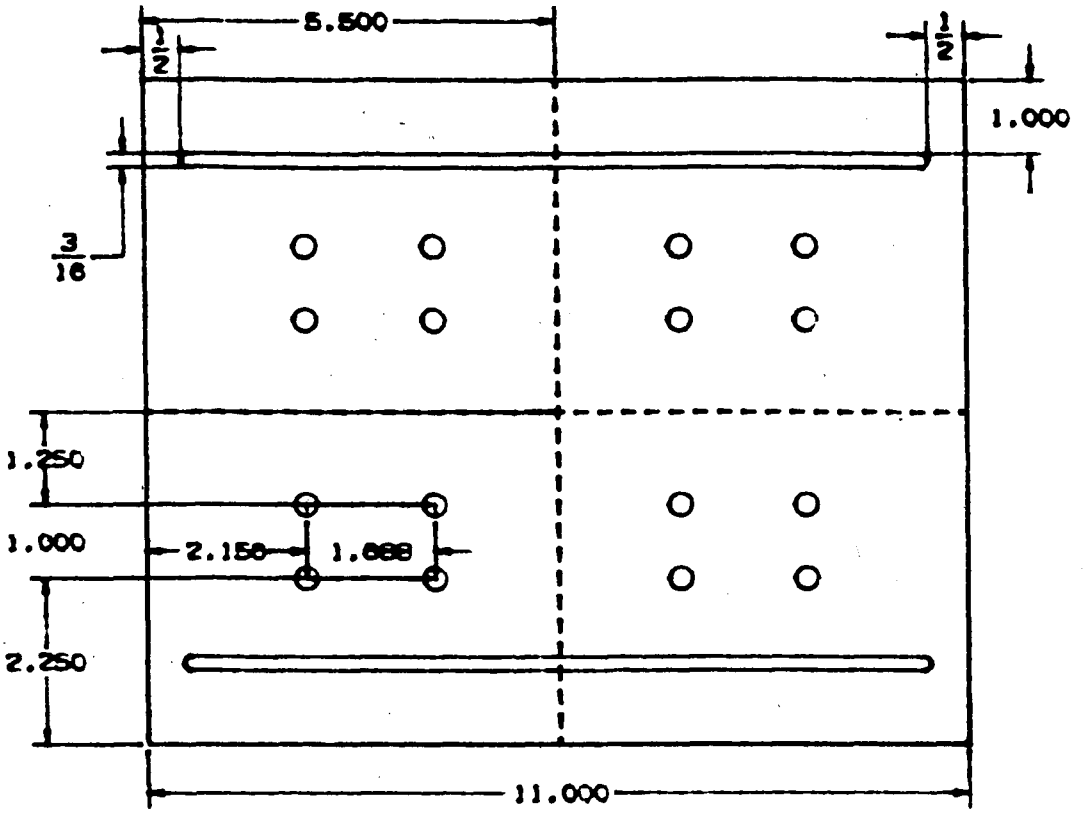
Figure 2.5b Tool Grip



Top View

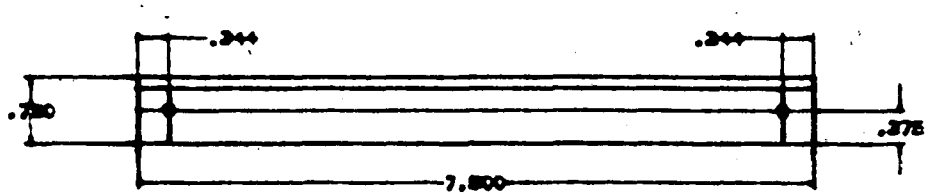
Side View

Figure 2.5b Tool Grip



Top View

Figure 2.6 Top Plate



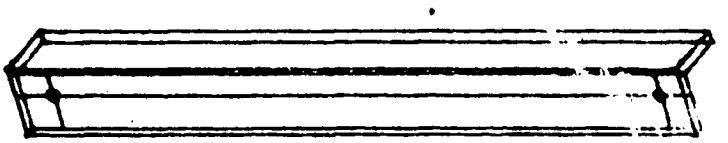


Figure 2.7 Flange to hold workpiece

Chapter III

THE INTERFACE

This is an important component of the system, and as the name implies does the job of interfacing the stepping motors to the computer. One can think of the computer as the brain, the controller software as the brain matter, the interface as the system of nerves, and the machine as the body.

The interface used for this machine consists not only of the circuit to amplify the signals outputted by the computer, but also of a " translator module " ie. a module that helps the programmer to input a pulse to it, and get a series of pulses out to turn the stepping motor one step angle. This is it's most important feature.

The following is a piece by piece description of it's components: (Refer Figure 3.1 for a diagram showing the general layout.)

Note: There is one such interface for each motor

The STM101 stepping motor translator module:

The STM101 is a printed circuit board containing the logic elements for bi-directional control of a SLO-SYN stepping motor. The unit can be triggered by pulses from an external pulse source, or an internal oscillator. In this case, the former is done ie. an external pulse source. A

22-pin edge connector, a 500K ohm speed control potentiometer, and a suppression network consisting of a 1 ohm, 5 watt resistor and a 10 mfd. capacitor are supplied. A separate 24vdc power supply is also required.

The mounting of the board has to be such that maximum convective flow of air is ensured. The unit comes equipped with it's own heat sinks.

Now, since in this case, an external pulse source viz. the MINC mini- computer is going to be used, it is important to see how the board is to be set up for this purpose. The potentiometer used to select the speed of operation is not needed. Also, the circuit has to be hooked up as shown in Figure 3.2. The triggering requirement for the STM101 board is as follows: Triggering is accomplished by a change in voltage at the "CW" pulse, or "CCW" pulse terminal from logic level 1 to logic level 0. Details are given below:

Logic 1 level-2.4 to 5.5 VDC

Logic 0 level-0. to 0.4 VDC

Minimum pulse width-10 microseconds

Maximum Fall time-1 microsecond

Maximum Rise time-10 microseconds

Loading-5mA maximum. With an open collector TTL device;a 4.7K ohm +/-10% pull-up resistor to 5 volts is required.

In the connection diagram (Figure 3.2) and throughout the text, direction of motion is referred to as either clockwise (CW) or counterclockwise (CCW). Each of these terms define the direction of motor shaft rotation as determined facing the name plate end of the motor.

The Power Resistors:

These are the dropping resistors as specified for the board, as well as the motor. For the larger M112-FD09 type motors, they are 6 ohm +/- 5%, 100 Watt resistors. They fit into the circuit as shown in Figure 3.2. For the M062-FD03 type motors, the ones required are 11 ohm +/- 5%, 50 Watt resistors.

The power supply:

The regulated constant voltage power supply described herein is designed to furnish regulated d.c. voltage. Within the limits of the specifications, this regulated supply delivers regulated voltage despite changes in input line voltage, line frequency, load impedance and temperature.

The power supply consists of a constant voltage transformer, a rectifying means, and a filter circuit. The transformer not only converts input line voltage, but is also the means of regulation. The rectifiers, by use of conventional circuitry, convert the AC to DC. The filter

circuit reduces the magnitude of the ripple to the desired specification level.

The nominal output voltage tolerance is +/-1% at nominal input voltage and full load. The output voltage is regulated to +/-1% over an input line variation of 100 to 130 volts RMS.

The power supply cannot be operated at an ambient temperature greater than 50 degrees C, or stored at a temperature greater than 85 degrees C. Refer Figure 3.3 for dimensions, and input/output connections.

The need for a regulated power supply arised from the specifications of the STM101 translator module. Although there were alternatives to this particular module, for economical reasons mainly, this one was chosen. Each of the larger motors need to draw approximately 6 Amps of current, and thus a " high power " power supply such as the one above was needed. The alternative was to go with three independent power supplies for each of the three motors. This was not economically a good idea, and therefore it was deceided that this particular power supply should be used, and the three motors hooked up to it in parrallel.

The port - NDRV-11:

This is a general purpose parallel interface used to connect TTL or DTL devices to the LSI-11 bus. The interconnection cable can be upto twenty five feet long. The interface permits program controlled data transfers upto forty thousand words per second. It also permits LSI-11 bus interface and control logic for interrupt processing and vector generation. The input data lines are protected by 16 diode-clamped input lines, and the output is handled by 16 latched output lines. Device addressing is assigned by the user. Control/Status registers and data registers are compatible with PDP-11 software routines. The NETCOM NDRV-11 is a direct replacement for the DRV11 module.

Specifications:

Identification: NDRV-11

Size:Double (22.8 by 13.2 cm.,8.9 by 5.2 in.)

Power:5.0 VDC +/- 5% @ 0.9 A

The mounting of this interface is done while taking into consideration the importance of the " portability " of this machine. The power supply stands by itself, and the three stepping motor boards are mounted inside a box, by using threads, and nuts to hold them in place. The connecting of the various parts is done while keeping in mind that the box (containing the stepping motor boards) is an independent unit, and with all the connections disconnected should remain independent.

The translator module is set up so that it responds to external pulses. The input to the module is to two data lines. Depending on whether the motor is to be turned clockwise or counter-clockwise, a pulse is fed to the appropriate input line of the module. This produces a turn in the corresponding direction.

The output of the module is a pattern of pulses that turn the shaft of the stepping motor by the angle corresponding to one step.

As the internal accelerator availible on the STM101 module is not being used, the speed of the pulses is controlled by the delay in pulsing the input of the module. Thus, the potentiometer is not hooked up to the module, and in it's place, the speed is directly controlled by the MINC mini-computer. This is done so as to give the programmer better control over the speed. This also saves one the trouble of hooking up another interface line to control the speed externally.

The major advantage of the interface used is obvious. The programmer has only to worry about setting one bit on his output port, to turn the motor clockwise or counter-clockwise. One does not have to worry about keeping track of the pattern of pulses necessary to turn the motor.

This eliminates the need to bookkeep the table of pulses in a stack in the computer, and then output them in an appropriate fashion. Also, signal amplification is taken care of by the interface.

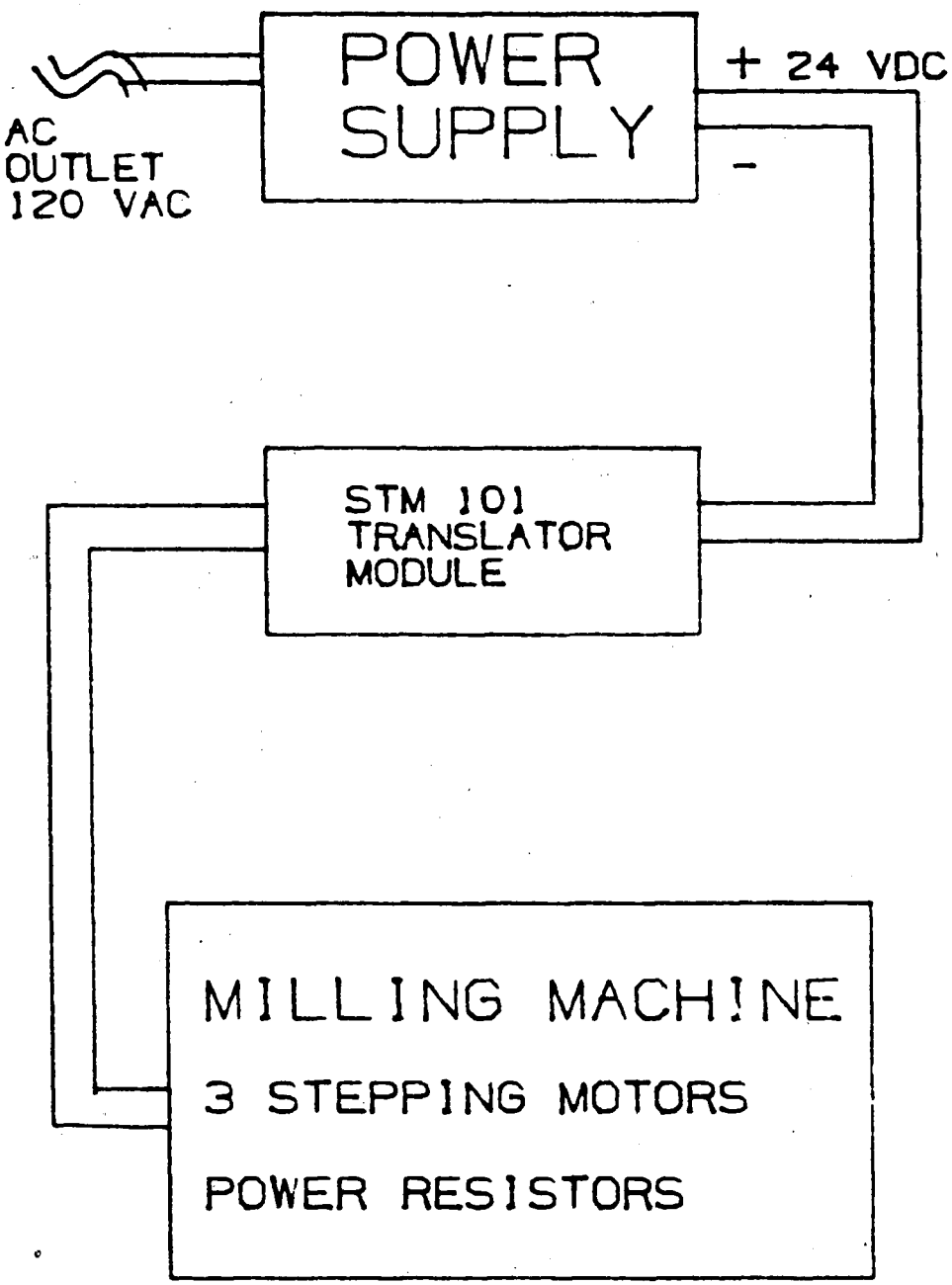


Figure 3.1 Layout of Machine/Interface

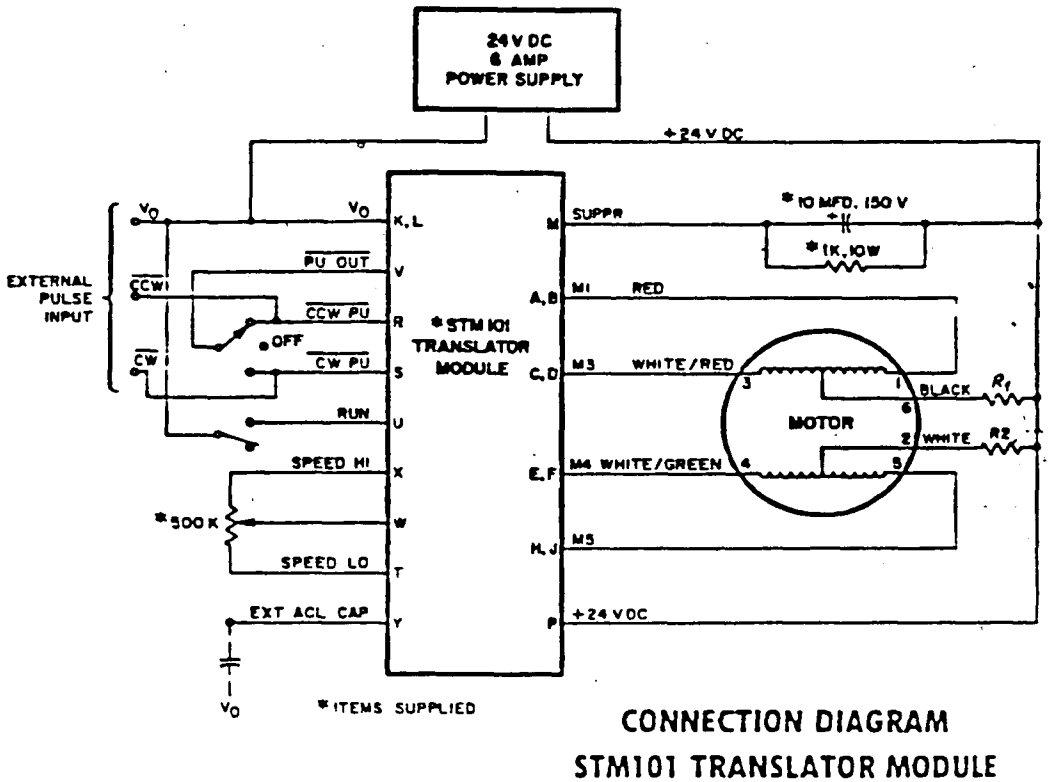
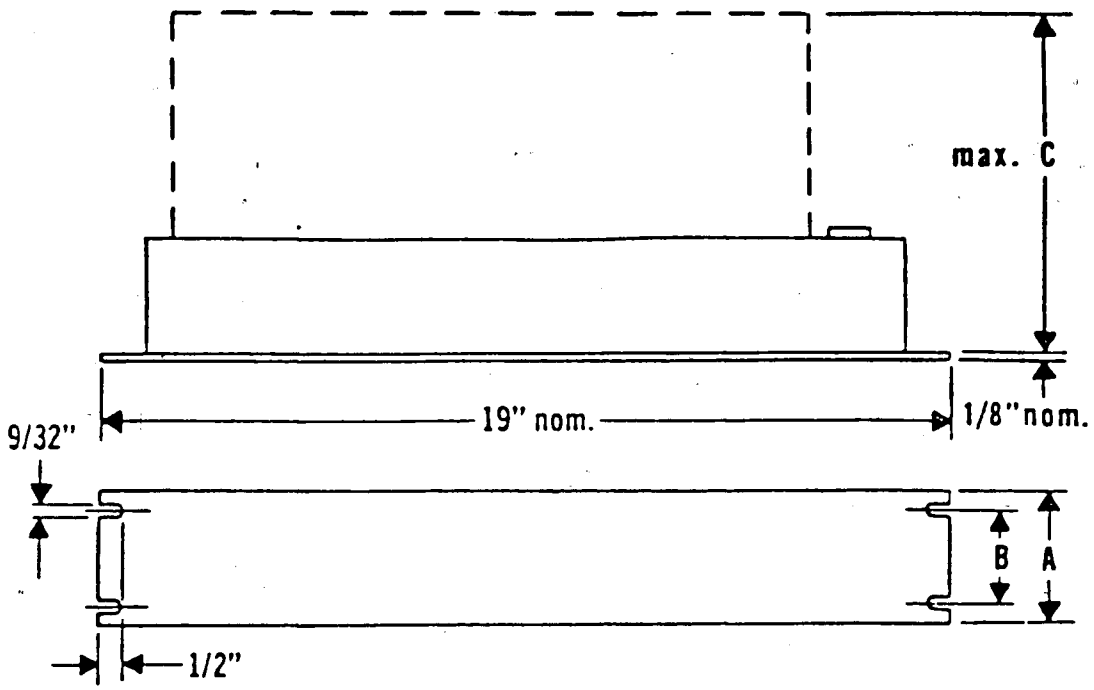


Figure 3.2 Interface details- hook up.



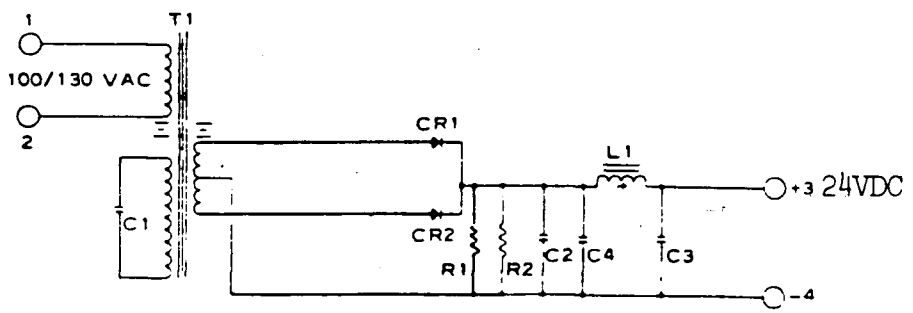


Figure 3.3 Power Supply Frame, and I/O.

CHAPTER IV

THE POST-PROCESSOR

This is a program written in FORTRAN which is responsible for converting the data generated by the NC machining simulation into data that can be used by the NC milling machine. This program is called "ANAND".

The NC machining simulation generates a file known as the CLSF (Cutter location source file), which contains all the information one needs to mill a part. However, the format of this file leaves a lot to be desired. The job of the post-processor (POST) is to convert this format into one which is at a level which can be understood by the milling machine's controller.

The final data generated by this particular post contains data which gives information on choice of motor, number of steps, and direction. This is the lowest level of data that can be generated by the POST. The main reason for doing this was to have the more powerful VAX 11/780 mini-computer do the crunching of data, and make the job easier for the less powerful (slower) MINC mini-computer. The following describes the software package in detail, giving information about input/output data, and the conversion routines.

The Input data:

As stated earlier, this is the CLSF. This data is generated by the NC simulation on the UNIGRAPHICS system. A listing of a sample CLSF follows, and each section marked on the listing is explained following the listing.

The Output data:

The output of this POST is such that the milling machine's controller can understand it. It contains choice of motor, the number of steps that the motor has to take, and the direction that the motor has to turn ie. clockwise, or anti-clockwise. Now, two lines of the CLSF can imply a lot of lines in the output of the POST. The lower level of this data is reason enough for that. A sample listing of part of the data output of the POST generated by the CLSF listed above is given below. The explanation of each marked section is also given.

Before explaining each routine and sub-routine of the POST, it is important to note the organization of the POST. The POST has been divided into three parts for a variety of reasons. Any post-processor is machine dependant, and specific to the machine that makes use of it's output data. The first two parts of the POST however were written with the intention of making it as general as possible so that other processors could make use of data generated by these parts of the POST.

This is the main reason for the break up of the POST into three different processors. The machine dependancy of the POST is introduced only in the third part of the POST or POST3. Another important reason for the break up is the organizational benefit that is acheived by this. A major factor that influences all this is that the author was responsible for not just the POST, but the controller as well. As both tasks proceeded simultaneously, it was easier to get the POST to generate a certain format, and then see if the format was suitable from the controller's point of view. If not, the data had to be processed further, thus resuling in one more part for the POST.

All POST1 or the first part of the POST does is break up the CLSF into three files, each of which contains statements that are similar. A fourth file has to be kept in order to remember the order of the statements. This part of the POST also removes the non-numerical data associated with any "GOTO" or "CIR" type statements ie. it converts all motion type statements into purely data statements.

The four files that are generated are listed below, with a brief explanation of it's contents.

GOTO.DAT- This file contains all the data associated with " GOTO " type statements contained in the CLSF. This means that it contains the co-ordinates of the point location contained in the "GOTO" statements.

CIR.DAT- This file contains two types of data - one which contains the co-ordinates of a location, and the second which contains the data associated with a "CIR" statement. The data arrangement is as follows. The co-ordinates of the point that the tool has to reach before encountering a "CIR" statement are recorded, after which data such as circle center, radius, normal axis etc. (contained in the CIR type statement) are noted. The final line of data gives the co-ordinates of the point that the tool has to reach after the end of the circular part. Therefore, for each "CIR" statement, there are three lines of data that POST1 generates.

MESS.DAT- This file contains all the statements in the CLSF that can be classified as "message" statements. They are like the ones indicated in the list of the sample CLSF file given earlier.

IPOIN.DAT- This file contains data that keeps track of the order of the various statements that are separated into the various files. All it contains is numbers one, two, and three. One implies that the next statement is contained in the GOTO.DAT file, two implies CIR.DAT, and three implies that MESS.DAT is the file that is to be accessed next. This is basically a file that does the book-keeping job for whatever software that is going to access the output of POST1.

The software along with the sub-routines will be discussed following a short description of the other two parts of the POST.

POST2 is responsible for going one step further along the data processing line. It eliminates the need for the book-keeping file, by condensing the data contained in the files generated by POST1, into a single file. This it accomplishes by adding a prefix to the data lines that are in the output of POST1. Also, it breaks up the high level of data contained in the output data CIR.DAT into a lower level of data. It makes use of information such as circle center, circle radius, and start and end point to generate the co-ordinate locations of the points that have to be accessed. It coalesces this data into GOTO.DAT, while maintaining the order of points to be accessed (using the file IPOIN.DAT). The message information contained in the file MESS.DAT it also puts into its one output file viz. POST2.DAT.

Hence, the general format of the file POST2.DAT is as follows: There are three basic statement types:

1) G,co-ordinates of point to be accessed.
2) C,co-ordinates of point to be accessed.
3) M,message to be displayed.

The only difference between statement 1), and 2) as indicated by the "G", or "C" is that the former has been derived from the "GOTO" statement, while the latter has been derived from the "CIR" type statement.

Now, POST3 was written to make the job of the controller even simpler. The input to POST3 is POST2.DAT, and it's output is POST3.DAT. The data contained in the output has been listed in the Appendix (Manual) and will be discussed later. It is more important to realize the objective of POST3 and how it is achieved. As stated earlier, one important factor in the writing of the POST was to maximize the use of the VAX and minimize the use of the MINC in making time consuming calculations. All the data necessary for the machine to make use of was already in an appropriate format for the machine's controller to respond to. All that was left was to make the controller decide on the number of steps, direction and motor choice. For the reasons stated earlier, POST3 was written for the sole purpose of reducing the burden on the controller.

Changing data representing straight line motion along either the X,Y, or Z axis was very easy. All that had to be done was that the relative displacement was to be divided by the displacement achieved by each step to yield the number of steps. The motor choice was obvious (depending on the axis). Depending on the sign of the relative displacement,

the direction (clockwise or anti-clockwise) was determined.

Contouring in the X,Y plane was relatively difficult. The sub-routine responsible for acheiving this is dealt with in detail at the end of the chapter. The sub-routine is called CONTOUR.FOR. The output of this POST3 is down to the basic motion level. It contains all the information needed for the controller to appropriately move the actuators (stepping motors). Now, for the contouring to be acheived, for two reasons it was unadvisable to break the data down into it's lowest level as in the case of the straight line motion along the X,Y, or Z axis. The first reason was that there were two motors that were involved, and had to be turned in such a way that although the number of steps for each was different, both had to reach their destination at the same time (i.e. they had to stop turning at the same time). Refer Figure 4.1. Hence, it was more important that a line of data be such that it contained a code to indicate that contour motion was involved, and also the number of steps that had to be taken by each motor were indicated along with their direction. The second reason for doing it the way it was done was that it cut down on the number of lines that the POST would have to output in order to specify just one motor at a time.

Now, message commands were handled a little differently. Those messages such as "COOLANT ON/OFF" etc. which were not really relevant to this small machine were ignored, and simply left alone to be displayed on the screen of the MINC by the controller. However, the message which indicated feedrate was handled differently. Rigid speeds were fixed (not variable), depending on the what range the feed rate lied in. Based on this, the delay in the controller was modified. This means that the time wasted between steps was changed.

What follows now is an indepth look at the sub-routines that make up this post-processor.

POST1.FOR:

Reads from the file AVI1.DAT, and writes it's output to the files: GOTO.DAT, CIR.DAT, MESS.DAT, and IPOIN.DAT. It reads the input file as characters. AVI1.DAT contains the CLSF file. The one parameter it assigns is IPOIN. The different values it takes are:

IPOIN = 1 - "GOTO" type statement implied.

IPOIN = 2 - "CIR" type statement implied.

IPOIN = 3 - "MESSAGE" type statement implied.

POST2.FOR:

Reads from the four files generated by POST1.FOR, and writes to the file POST2.DAT. Part of this routine is responsible for converting the data contained in the CIR.DAT file, and converting it into simple co-ordinates of points. It does this by first determining the quadrant that the first point that the circle passes through is in with respect to the center of the circle at the origin. Then using the formula for the tangent of an angle, and assigning the appropriate sign to the angle, it computes the x, and y co-ordinates of the point (with respect to the center of the circle as the origin). Then, it adds these to the work co-ordinate system of the part. Four of the possible cases are given in Figures 4.2a thru 4.2d.

This is the major job of POST2.FOR. The other task of putting a prefix before the co-ordinate data is merely a question of changing the format. One final thing that it does is that it reads the message statement, and puts a prefix before the statement (M), before writing it out to the output file.

POST3.FOR:

This part of the POST as indicated earlier, relieves some of the job of the controller as well. Unlike the other parts, this one makes use of lower level routines,sub-routines to reduce the input data to machine level commands. What follows is a look at the various sub-routines that make up this part of the POST.

SUBROUTINE MOT1.FOR:

This routine is responsible for determining whether contour motion is necessary or straight line motion is necessary. It assigns qualifying parameters depending on the type of motion desired. The parameter assigned is 'ICODE', and it can take the following values:

ICODE = 11 - STRAIGHT LINE MOTION ALONG THE X AXIS.

ICODE = 12 - STRAIGHT LINE MOTION ALONG THE Y AXIS.

ICODE = 2 - VERTICAL MOTION ALONG THE Z AXIS.

ICODE = 3 - CONTOUR MOTION IMPLIED.

All this routine does is that it looks at two successive points that the machine tool has to access, and compares them. It does the comparing by subtracting each respective co-ordinate, to get "DETLAs" or differences in each direction. Depending on them being zero or not zero, the descision is made.

SUBROUTINE MOTSX.FOR:

Once it is determined that the motion is straight line motion along the X axis, then this routine does the job of simply computing the direction, and the number of steps that the machine has to take. It makes use of the ratio of one step to distance travelled for the "X motor". The direction, it determines by simply noting the sign of the displacement. The choice of the motor has been pre-determined. It then writes this data as output in the file POST3.DAT. The format of the data is as indicated earlier in the listing of the data file.

SUBROUTINE MOTSY.FOR:

This routine does the same thing as MOTSX.FOR, except that the choice of motor is now the "Y motor".

SUBROUTINE MOTV.FOR:

Again, virtually the same thing as done in the case of motion along the X axis is repeated. However, the format of the data statement is a little different. The choice of motor is obviously the "Z motor".

SUBROUTINE CONTOUR.FOR:

This is probably the only routine in this software that does any conversion of data, except for the routine in which data associated with the "CIR" statement is converted into point co-ordinates. All this routine does is use a "step"

method to go in a straight line from one point to another point. This is done in the case where the straight line is not aligned along either of the three principal axes - X,Y, or Z. The algorithm for the "step" approach used is on the following page. This routine directly writes it's output to the output of this POST viz. POST3.DAT. The data line that this routine outputs contains the character C to indicate contour motion in the X,Y plane, number of steps for the "X motor", number of steps for the "Y motor", and direction for each motor. The format of this data line is given earlier in the description of the output data of the POST.

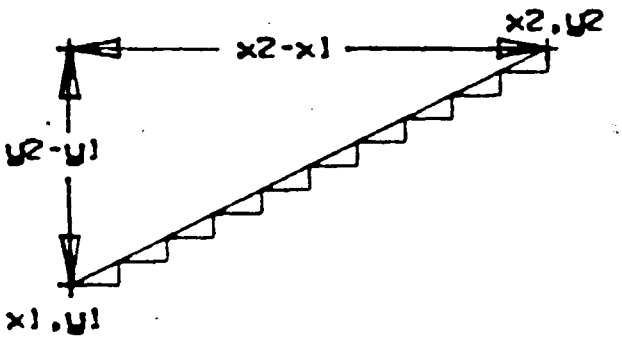
The final function of this POST is to assign the delay parameter (mentioned earlier) that determines the speed of the stepping motor. This depends solely on the feed rate given in the CLSF. This is the only message type statement that POST3 processes. All the others it leaves alone for future display on the MINC's terminal.

As evident from the description of the POST, espescially POST3, there is a lot of interaction between the controller, and the post-processor. Basically, since their function is ultimately one of processing data in order to control the milling machine, therefore they have to go hand in hand. The basic advantage of having the post- processor, is that instead of having a wide variety of data files, which are produced for the wide variety of milling machines,

only one data file (such as the CLSF) needs to be generated. Then, a post-processor that is applicable to a particular machine can be used effectively to change the format of data. This is what makes the post-processor machine dependant. Another major advantage to having the post-processor is that in case any mistake is detected in a complicated machining process, then changing it is relatively easier. The correction can be done at the post-processing level instead of having to re-do the entire machining operation.

For a listing of the Cutter Location Source File and Post-processor output, please refer Figures A.2, and A.3 respectively.

CONTOURING BASICS



OF STAIRS IN X = # OF STAIRS IN Y = N

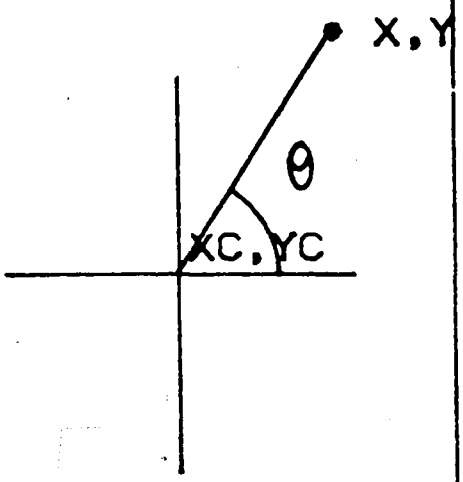
STEP FACTOR (S.F.) = INCHES/STEP

$$\text{RATIO (R)} = \frac{\text{\# OF STEPS FOR Y MOTOR}}{\text{\# OF STEPS FOR X MOTOR}}$$

R GT. 1 - 'X STAIR' = S.F.

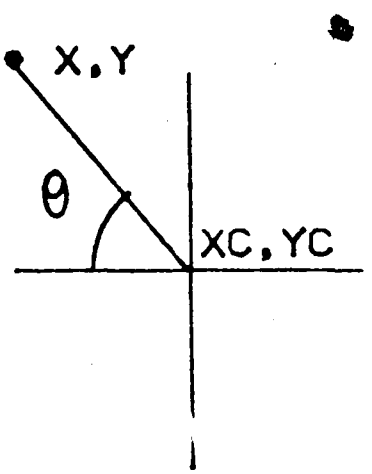
R LT. 1 - 'Y STAIR' = S.F.

Figure 4.1 Basic Contouring in the XY plane.



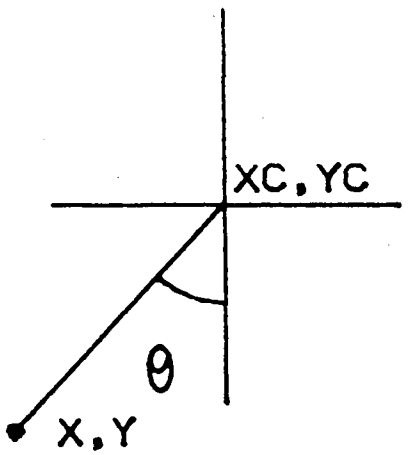
START ANG = θ

Figure 4.2a



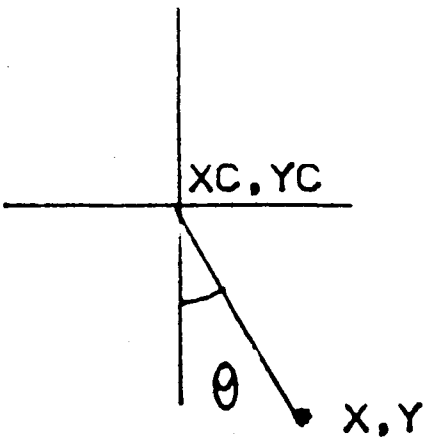
START ANG = 180-θ

Figure 4.2b



START ANG = 270-θ

Figure 4.2c



START ANG = 270+θ

Figure 4.2d

CIR Data - locations possible

Chapter V

THE CONTROLLER

The controller is the brain of this CNC mill. It consists of software that controls the milling machine. The program is called "REKHA". The input to this controller is the output of the post-processor (POST). The output of it is appropriate motion in the actuators ie. stepping motors. One can also consider the interface as part of the controller, but this is only a matter of semantics. Since the interface has already been dealt with in a preceding chapter, this chapter will deal with the software that controls the machine only. The host to the controller is D.E.C.'s MINC mini-computer (PDP11).

The software is divided into two halves on the basis of the language in which it is written. This implies that one half consists of higher level routines that do computing of various parameters, and the other half of lower level routines that actually drive the stepping motors using the parameters.

It is important to review the input data to the controller, or the output data of the POST, before one looks in detail at the software.

There are four basic types of statements:

1) 'S', data

2) 'C', data

3) 'F', data

4) 'M', data

In the first case('S' data), straight line motion is commanded and the data consists of choice of motor, number of steps, and direction. In the second case ('C' data)contour motion is implied, and the data gives information about the number of steps, and direction for each motor. For the exact format of the data output refer to the previous chapter in which a listing of the output data is present. Contour motion simply means that the motion is in the XY plane, and both the X, and Y motor have to be turned in order to get the right motion. In the third case('F' data), 'F' stands for feedrate, and the feedrate parameter is the data that is given. In the last case('M' data), data is the message that is to be displayed.

Keeping the above output data in mind, let us take a look at the software. Both the FORTRAN, and the MACRO-11 software are described below in detail.

A) FORTRAN SOFTWARE:

This part of the controller is responsible for sorting the output of the POST and sending the appropriate input to the MACRO level routine to drive the stepping motors. The best way to examine this routine is to look at each line of data input, and see how the routine handles it.

'S' data- Since straight line motion is implied, the only concern is about identifying the appropriate motor, and outputting the number of steps and direction to it.

'C' data- This is a little more difficult to handle. One needs to look at the Contouring algorithm again, in order to understand more clearly what has to be done. As indicated in Figure 4.1, knowing the ratio of the steps needed in the x direction, and those needed in the y direction one knows the ratio. Hence, along the axis in which the distance to be travelled is the lesser, the minimum increment is taken to be the step factor (inches/step), and the minimum increment along the other axis is the ratio of the steps (greater than the step factor).

'F' data- This simply indicates the speed at which the motor is to be run. The FORTRAN routine uses this information to set up the time lag between individual step of the stepping motors. For straight line motion, the feed

speed tends to be faster than when the tool is cutting along a curved path. Hence, the time lag in straight line motion is relatively short as compared to contour motion.

'M' data- The 'M' data is handled simply by the message being displayed on the screen.

This FORTRAN routine is also responsible for keeping track of the position of the machine with respect to the origin. For information on the set-up of the zero or home location, refer the USER's MANUAL in the Appendix. The reason that it is important to keep track of the location is that it is easy to return to the ZERO or HOME once this information is availible. This routine is discussed in detail after the discussion on the MACRO routine.

B) MACRO SOFTWARE:

This routine is responsible for responding to the input data given by the FORTRAN routine and driving the appropriate stepping motor via the STM101 interface. What is done is that the six possible combinations of data that can be sent to the STM101 are stored in the Computer's memory. (The six possibilities arise from there being three motors, and two possible choices for direction.)Then, depending on the FORTRAN routine's choice the appropriate data byte is sent to the parallel output board DRV11-J by a MOV command. This piece of data is held at the port for a

short time (about two milliseconds), and then a zero is sent to the port. This 1 to 0 transition triggers the STM101 to output the required table to the stepping motor. Hence, motion takes place in the machine. This is the main purpose of the MACRO routine. It also handles the number of steps specified by the FORTRAN routine by a simple loop method.

HOME LOCATION/ POSITION TRACKING:

There are two other sub-routines (written in FORTRAN)that are an additional feature of this Controller. One is called HOME, and the other is called TRACKE. Each time that the MACRO routine is called and the stepping motors are "moved", the TRACKE routine is also called.

Sub-routine TRACKE:

This routine keeps track of how many steps each motor has taken (Clockwise is taken to be positive, and Counterclockwise to be negative).

Sub-routine HOME:

Now in order to ensure that the user always starts the machine from the zero location, care has to be taken that there is an easy way for the user to start from zero. Hence, using the information provided by the TRACKE routine, the HOME routine retraces it's steps to the origin so that the next user does not have to bother about resetting the machine. At the end of the machining sequence, the user is

asked to remove his part before the machine returns to it's zero location.

Chapter VI

CONCLUSIONS and RECOMMENDATIONS

The objectives of this effort have been met. A computer controlled milling machine that responds to data contained in a file at the time of creation of the part (on a CAD System-UNIGRAPHICS) has been built.

The design of the machine demonstrates the great care that a designer of numerically controlled milling machines has to take in designing and fabricating such machines. Stepping motors have been chosen as actuators to provide the advantage of simpler control. By using ball screws for linear motion, a high degree of accuracy is obtained in the machine- upto .0012". The specification that the machine be portable has been met. The total unit is such that the machine and electronics are separate components that can be hooked up together very easily when the machine is to be used. The machine is comparatively light,about 100 pounds;it is easily moved by two people. The design specifications of motion have also been met (8" by 10" by 3").

Besides accomplishing the goal of designing and fabricating a genuinely portable NC milling machine, the other goal of providing the machine with true computer control has also been met. Data generated on the CAD system UNIGRAPHICS is post-processed appropriately on the VAX 11/780 , and a machine dependent file is generated. The post-processor used is ANAND. This machine dependent data is then input to the controller of the NC milling machine which drives the actuators of the machine in the appropriate fashion. The controller is called REKHA, and the machine it controls is called JIMMY.

Although this machine is a computer controlled machine in the fullest sense, it's control is open loop. There is room left for introducing closed loop control of this machine using optical encoders at the end of the ball screw. In the Mechanical Design, room has been left for encoders to be added. The NDRV-11 is already equipped with an input buffer, and, therefore the interface poses no additional difficulty in providing for closed loop control. Another avenue for development is the modifying of the controller for three axis machining. This would also imply changes in the post-processor. Also, the tool, and hence the tool axis would have to be modified in order to allow for 3-axis machining.

References

[1] Graphics Multi-Axis Module
MCAUTO, St. Louis, Mo, 1980.

[2] Graphics Machining Editor
MCAUTO, St. Louis, Mo, 1980.

[3] Graphics Mill Module
MCAUTO, St. Louis, Mo, 1980.

[4] Graphics Interactive Programming
MCAUTO, St. Louis, Mo, 1979.

[5] Unigraphics User's Manual
MCAUTO, St. Louis, Mo, 1980.

[6] Unigraphics Programming Manual
MCAUTO, St. Louis, Mo, 1979.

[7] Newman, William M. and Sproul, Robert F.
Principles of Interactive Computer Graphics.
McGraw Hill, 1979.

[8] Gill, Arthur.
Machine and Assrmbly language programming of the PDP-11.
Prentice-Hall,2nd. edition, New Jersey, 1983.

[9] Baumeister, Theodore.
Mark's Standard Handbook for Mechanical Engineers.
McGraw Hill, 8th. edition, New York, 1978.

[10] Thomson Ball Bushings for Linear Motion
Thomson Industries, Inc., Manhasset, N.Y.,1981.

[11] Slo-Syn Stepping Motors
Superior Electric, Bristol, Conn., 1979.

[12] Slo-Syn Stepping Motor Controls
Superior Electric, Bristol, Conn., 1979.

[13] PDP11 Software Handbook
DIGITAL, Maynard, MA, 1979.

[14] MINC Interfaces Handbook
DIGITAL, Maynard, MA, 1981.

[15] Spotts, M.F..
Design of Machine Elements.
Prentice-Hall, 5th. edition, 1978.

[16] Warner
Warner Electric, Illinois, 1982.

[17] Hall, Douglas V..
Microprocessors and Digital Systems.
McGraw Hill, New York, 1980.

[18] The Cincinatti Milling Machine CO..
A treatise on milling and milling machines.
Cramer-Krasselt, 3rd. edition, 1951.

[19] American Society of Mechanical Engineers
Numerical Control in Manufacturing
McGraw Hill, New York, 1963.

APPENDIX

USER'S MANUAL

The following text will provide the user with a step by step guide on how to use this machine. Before one delves into the details of it, it is important to note the following:

1) While this thesis was being written, the author was in the process of making improvements and changes to further the integration of the system. Hence, some changes have probably been made since this was written. The changes will be on record in the CAD/CAM program office.

2) It is assumed that the user is familiar with the NC machining module on the UNIGRAPHICS system.

In order that the user has a good idea of the entire process, a brief explanation of the integration is given below.(Those interested in knowing in detail the workings of this process are welcome to read the thesis that this documentation is a part of.

The objective of this system is to be able to create a part (that can be machined using a two and a half axis machine)on the CAD system UNIGRAPHICS, and be able to produce it on the NC machine that this system is a part of.

Once the user has created a part on the system like the one shown in Figure A.4, and has performed the NC simulation using the UNIGRAPHICS system, a file is created known as the Cutter Location Source File or CLSF (refer Figure A.2). This CLSF is post-processed by a post-processor known as ANAND. A part of the post-processed version is shown in Figure A.3. This file is the input to the brain of the milling machine ie. to the controller. The host for this controller is D.E.C.s MINC LSI-11 mini-computer. The controller software is called REKHA. The final output of the controller consists of binary data that drives the milling machine's actuators via the electronic interface. Hence the post-processing of data is a two-step process:

1) Post-processing of the CLSF

2) Feeding the post-processed version to the controller to drive the actuators.

What follows is a step by step guide for using the machine.

1) Keeping in mind the work area shown in Figure A.4, and the limits of motion, the user has to design his/her part on the UNIGRAPHICS system. The limits of motion are 10" by 8" in the XY plane. Also, he or she has to be careful in noting that the origin of the part is in as indicated in Figure A.1. This implies that no negative co-ordinates are accessible by the tool. Hence, the user should be careful

in choosing the work co-ordinates when creating the part. A simple check to see if the part creation has been correct, is a look at the CLSF's "GOTO" statements. If the co-ordinates associated with them are all positive, there is less likelihood of a mistake. Please make sure to note the location of the slots for the fixture that holds the workpiece in place. The fixture is shown in Figure 2.7.

2) On the UNIGRAPHICS system, generate the CLSF by performing the simulation.

3) Convert this CLSF to the Files-11 format by using either the "USER FUNCTION" key on UNIGRAPHICS or by executing the program UGU001 in the UNIGRAPHICS library.

4) Copy this Files-11 format of your CLSF into your directory by using the following command:

$COPY "DEVICE":["UNIGRAPHICS DIRECTORY"]"PART NAME" AVI1.DAT

This action should produce a CLSF in your directory by the name of AVI1.DAT. Look at it to make sure that it resembles the file shown in Figure A.2. Otherwise repeat steps 2) and 3).

5) Modify the file AVI1.DAT in the following manner:

a) A "GOTO" statement has to be added before the first "GOTO" statement is encountered, to make sure that the tool goes from the zero location to the starting point in a "proper" fashion. The type of statement that has to be added is shown in Figure A.2.

b) A close look at the CLSF is important at this point. The post-processor and controller for the system provide for two and a half axis machining.(This implies that motion is possible only along either the X,Y,or Z axis, or that contour motion is possible in the XY plane.) If during the simulation, the user has made a mistake in choosing the clearance plane etc., the CLSF will have certain sucessive "GOTO" statements that show all 3 X,Y, and Z axes changing. Now, instead of going back to the simulation and redoing it, the user can insert statements(of the "GOTO" type) in the CLSF so as to ensure two and a half axis machining. Refer Figure A.2.

6) Run the globally accessible program called ANAND by simply typing ANAND. A file by the name POST3.DAT should appear in the user's directory.

7) Write this file onto a floppy disk. See the system manager about how to do this.

8) Input the disk on drive 2 of the MINC mini-computer. Boot the MINC by turning it on while making sure that the disk labelled BOOT MILL is in drive 1.

9) Turn on the milling machine, and make sure that it is connected to the MINC via the DRV11-J.

10) Mount the part on the machine making sure that the co-ordinate systems match up as indicated in Figure A.4.

11) Make sure that the machine is in it's home position as indicated in Figure A.4. Otherwise see the system manager.

12) On the MINC's keyboard type: RUN REKHA. Respond to the self-explanatory prompts that the software provides you with appropriately. Make sure that after removing the part you respond to the prompts in such a way that the tool's final position is "home".

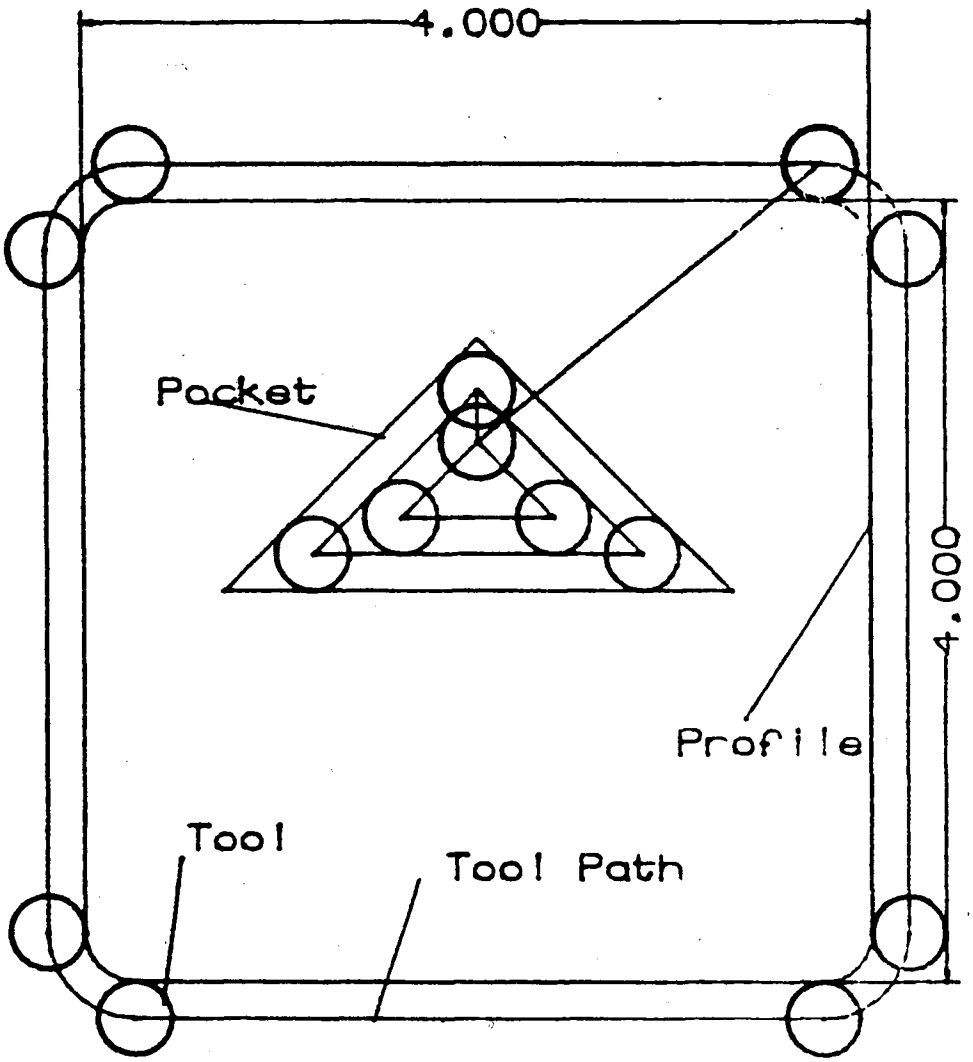


FIGURE A.1 A SAMPLE PART

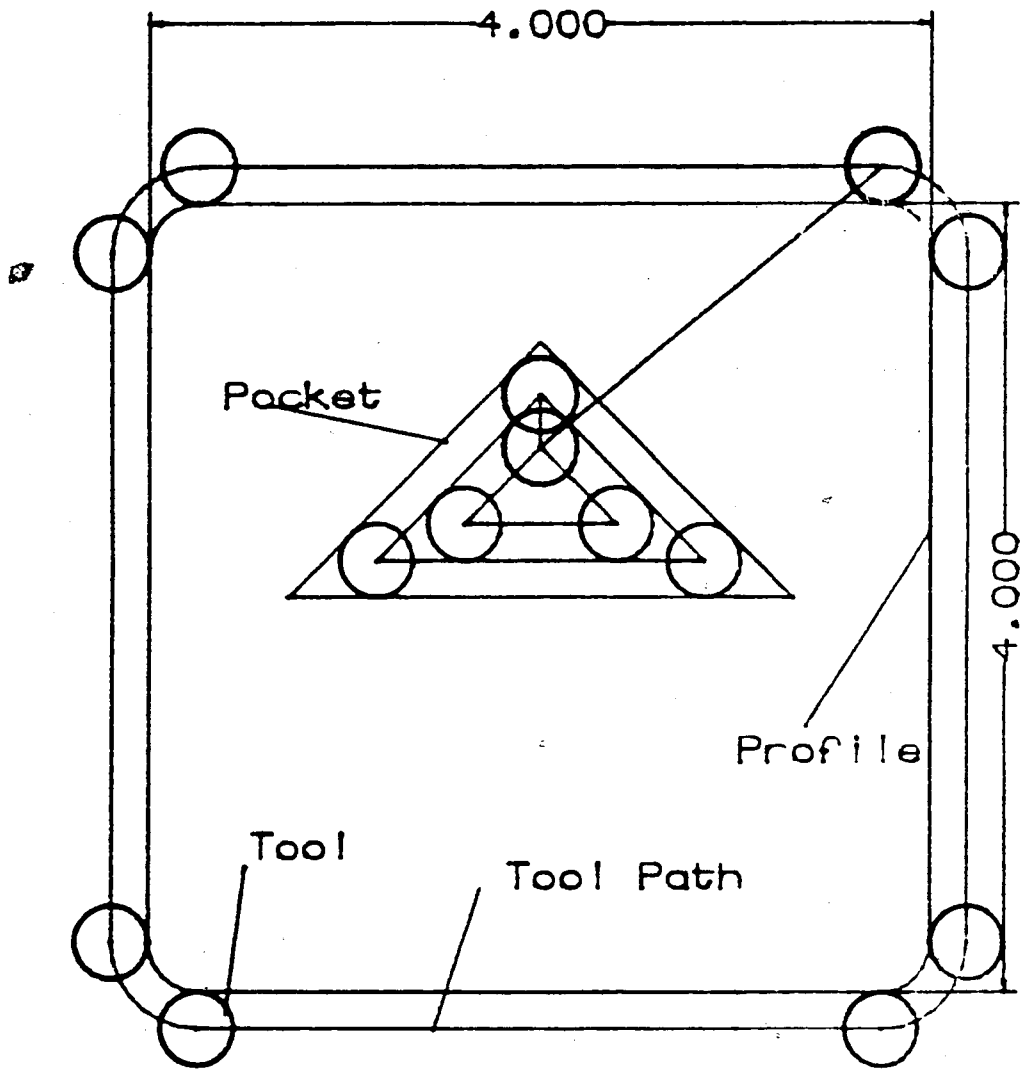


FIGURE A.1 A SAMPLE PART

```
TOOL PATH/0. 188, 0. 188, 1. 000, P4
MSYS/-4. 0000000, -4. 0000000, 0. 0000000, 1. 0000000,
RAPID
PAINT/COLOR, CYAN
GOTO/0. , 0. , 2. **      This line was manually added.    **
GOTO/5. 750, 6. 188, 2. 000, 0. 000, 0. 000, 1. 000
GOTO/5. 750, 6. 188, 0. 000, 0. 000, 0. 000, 1. 000
FEDRAT/10. 000
GOTO/2. 250, 6. 188, 0. 000, 0. 000, 0. 000, 1. 000
CIRCLE/2. 250, 5. 750, 0. 000, 0. 000, 0. 000, -1. 000, 0. 4
GOTO/1. 813, 5. 750, 0. 000, 0. 000, 0. 000, 1. 000
GOTO/1. 813, 2. 250, 0. 000, 0. 000, 0. 000, 1. 000
CIRCLE/2. 250, 2. 250, 0. 000, 0. 000, 0. 000, -1. 000, 0. 4
GOTO/2. 250, 1. 813, 0. 000, 0. 000, 0. 000, 1. 000
GOTO/5. 750, 1. 813, 0. 000, 0. 000, 0. 000, 1. 000
CIRCLE/5. 750, 2. 250, 0. 000, 0. 000, 0. 000, -1. 000, 0. 4
GOTO/6. 188, 2. 250, 0. 000, 0. 000, 0. 000, 1. 000
GOTO/6. 188, 5. 750, 0. 000, 0. 000, 0. 000, 1. 000
CIRCLE/5. 750, 5. 750, 0. 000, 0. 000, 0. 000, -1. 000, 0. 4
GOTO/5. 750, 6. 188, 0. 000, 0. 000, 0. 000, 1. 000
GOTO/5. 750, 6. 188, 1. 000, 0. 000, 0. 000, 1. 000
END-OF-PATH
TOOL PATH/0. 188, 0. 188, 1. 000, P5
MSYS/-4. 0000000, -4. 0000000, 0. 0000000, 1. 0000000
FEDRAT/10. 000
PAINT/COLOR, CYAN
GOTO/3. 612, 4. 375, 1. 000
GOTO/3. 612, 4. 375, 0. 000
GOTO/4. 000, 4. 763, 0. 000
GOTO/4. 388, 4. 375, 0. 000
GOTO/3. 612, 4. 375, 0. 000
GOTO/3. 160, 4. 188, 0. 000
GOTO/4. 000, 5. 028, 0. 000
GOTO/4. 340, 4. 188, 0. 000
GOTO/3. 160, 4. 188, 0. 000
END-OF-PATH
```

Figure A.2 A Cutter Location Source File.

```
M TOOL PATH/0.188,0.188,1.000,P4
M MSYS/-4.0000000,-4.0000000,0.0000000,1.0000000,
M RAPID
M PAINT/COLOR,CYAN
S2 16670
C 47920 51570
S2 16671
F1
S0 29171
C  1261   221
C  1101   641
C   811   971
C   431  1191
C    51   621
S1 29171
C   220  1261
C   640  1101
C   980   811
C  1190   431
C   620    41
S0 29170
C  1270   220
C  1100   640
C   810   980
C   430  1190
C    50   610
S1 29170
C   221  1250
C   641  1100
C   971   820
C  1191   430
C   631    50
S2  8330
M END-OF-PATH
```

Figure A.3 Post-processor Output.

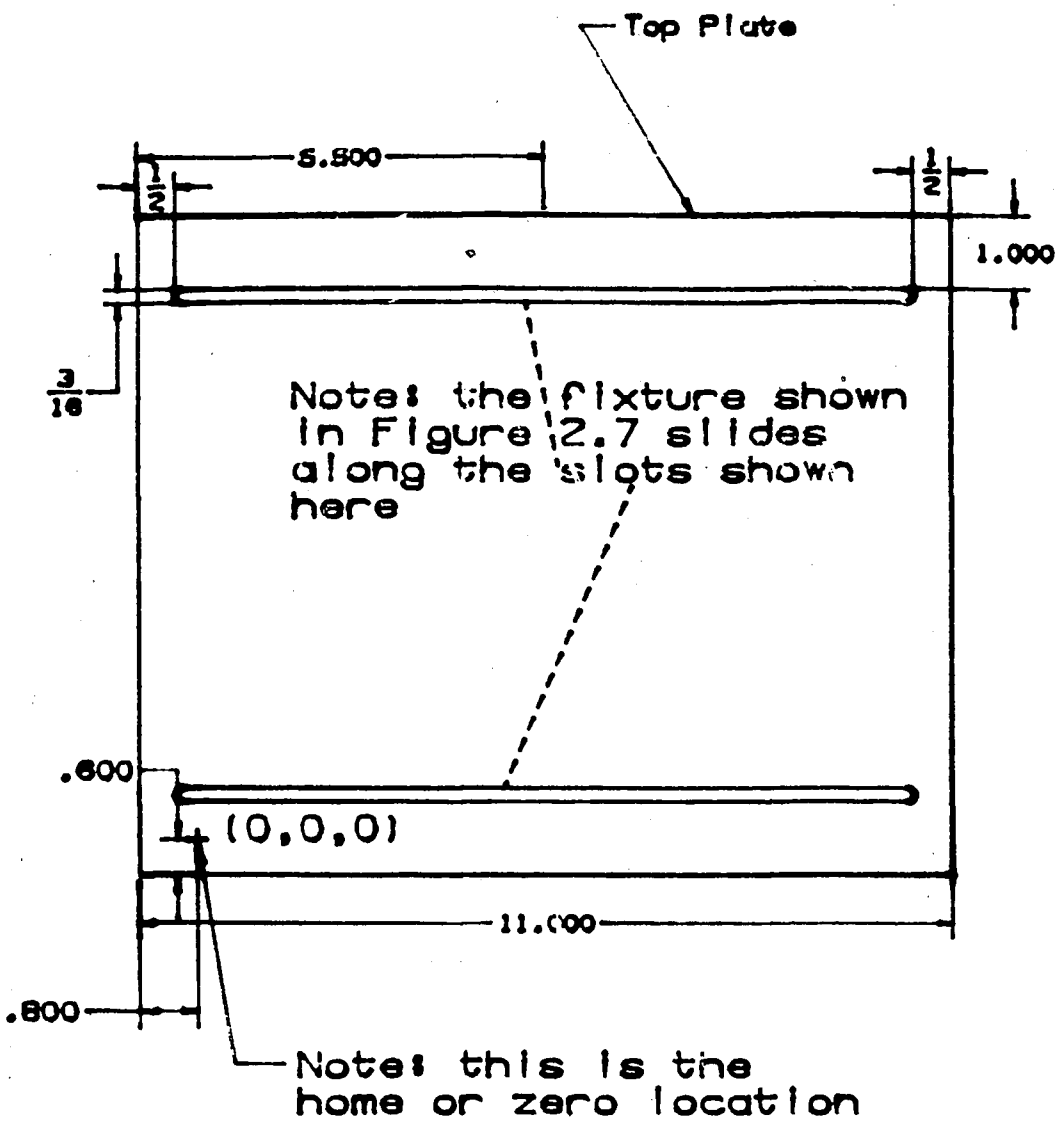


Figure A.4 Set-up of work area

VITA

The author was born in Bombay, India on January 15th, 1962.

He attended Howard University in Washington D.C. from August 1978 to May 1982. He obtained a B.S. magna cum laude in Mechanical Engineering. While at Howard he was made a member of the National Engineering Honor Society - Tau Beta Pi.

He began graduate school in September 1982, working towards a Master of Science in Mechanical Engineering at Lehigh University. During his stay at Lehigh, he worked as a Research Assistant for the CAD/CAM program.

His parents are Mrs. Rekha A. Kenkare and the late Mr. Anand B. Kenkare.